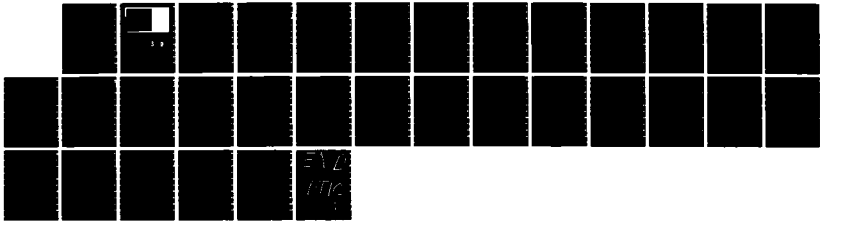
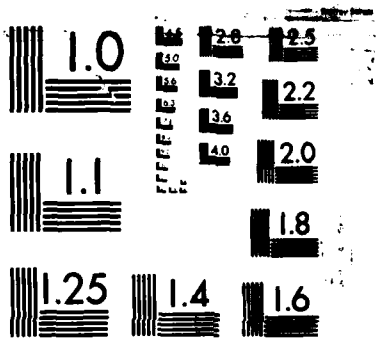
AD-A167 491 COMPUTING NARROW INCLUSIONS FOR DEFINITE INTEGRALS(U) 1/1
WISCONSIN UNIV-MADISON MATHEMATICS RESEARCH CENTER
G F CORLISS FEB 86 MRC-TSR-2913 DAAG29-80-C-0041
UNCLASSIFIED F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS

MRC Technical Summary Report #2913

COMPUTING NARROW INCLUSIONS FOR DEFINITE INTEGRALS

George F. Corliss

Mathematics Research Center
University of Wisconsin—Madison
610 Walnut Street
Madison, Wisconsin 53705

February 1986

(Received February 20, 1986)

Sponsored by

U. S. Army Research Office
P. O. Box 12211
Research Triangle Park
North Carolina 27709

UNIVERSITY OF WISCONSIN-MADISON
MATHEMATICS RESEARCH CENTER

COMPUTING NARROW INCLUSIONS FOR DEFINITE INTEGRALS

George F. Corliss

Technical Summary Report #2913
February 1986

ABSTRACT

We use ACRITH to implement algorithms which give narrow inclusions for definite integrals. Inclusions are obtained from familiar numerical quadrature formulas such as Gaussian or Newton-Cotes with remainder terms, or from the term-by-term integration of Taylor polynomials with remainder terms. Inclusions for the remainder terms are computed using automatic differentiation. The inclusions are valid if the integrand or the endpoints of the interval of integration are real- or interval-valued. Interval inclusions which contain only a few machine numbers are achieved by using ACRITH's accurate scalar product, by using order and subinterval adaptation, and by using special devices such as intersection of several estimates. Numerical examples show that such narrow, *guaranteed bounds* require about four times as long to compute as the *estimates* computed by the routine QAGS from QUADPACK.

AMS (MOS) Subject Classifications: 65D30, 65G10

Key words: Adaptive quadrature, self-validation, ACRITH, interval arithmetic.

Work Unit Number 3 (Numerical Analysis and Scientific Computing)

Sponsored by the United States Army under Contract No. DAAG29-80-C-0041.

SIGNIFICANCE AND EXPLANATION

Routines for numerical integration are among the fundamental building blocks of scientific computation. They are typically called from deep inside an applications program where the user is rarely able to examine the results. Commonly used quadrature routines such as CADRE and QUADPACK have proven to be reliable and robust. However, they compute only an *approximation* to the correct answer and an *estimate* for the error in the value returned. In principle, they can be completely wrong.

The program described in this report performs self-validating quadrature to return an interval in which the correct answer is *guaranteed* to lie. Gaussian, Newton-Cotes, and Taylor polynomial quadrature formulas with remainder terms are captured using interval computations provided by ACRITH running on IBM 370 class mainframe computers. The width of the interval which is guaranteed to contain the answer can often be made as small as the user wishes, or as small as an interval containing only a few machine numbers.

It costs about four times as much CPU time to find a guaranteed answer as it costs to find an estimate.

The responsibility for the wording and views expressed in this descriptive summary lies with MRC, and not with the author of this report.

COMPUTING NARROW INCLUSIONS FOR DEFINITE INTEGRALS

George F. Corliss

1. Introduction.

We wish to compute

$$If = \int_A^B f(x)\,dx. \tag{1.1}$$

That is, given the input information:

Limits:	Finite limits of integration A and B (perhaps intervals).
Integrand:	A Fortran-like expression for the integrand f as a finite sequence of $+, -, *, /, \sqrt{}$, exp, ...
Tolerances:	α and ρ, tolerances for the desired absolute and relative errors.
Evaluations:	A limit M on the number of function evaluations allowed.

We wish to compute an interval $J = [c, d]$ satisfying the goals:

Inclusion:	$If \in [c, d]$.
Accuracy:	$w(J) = d - c \le \max\{\alpha,\ \rho \cdot \lvert If \rvert\}$.
Bounded cost:	At most M function evaluations are used.
Efficiency:	The computations are as efficient as possible.

The algorithms described here nearly meet these goals by using information about the continuous set of values of the integrand. In contrast, standard methods use only a finite set of values of f and can be fooled badly.

The goal of inclusion is met by this program provided only that f can be evaluated at every point on the interval of integration. Otherwise, the program notifies the user that the integrand cannot be evaluated.

Sponsored by the United States Army under Contract No. DAAG29-80-C-0041.

The goal of accuracy is met provided that the accuracy requirement specifies an interval which contains at least a few machine numbers, and that the integrand is not too badly oscillating. If the requested accuracy cannot be met, the program returns the narrowest interval it can compute subject to the bounded cost constraint, along with a warning that the requested tolerance has not been met.

The goal of bounded cost may be relaxed by about 10%. When the maximum number of function evaluations is reached, the computations which are required on the current subinterval are completed. This results in a slight cost over-run, but function evaluations which were already made are not wasted.

The goal of efficiency is more nebulous. Overall, the program computes a guaranteed inclusion in about four times the CPU time required for the QUADPACK routine QAGS [14] to compute an estimate. We have observed CPU times as slow as 50 times QUADPACK's time, but there are problems such as those with polynomial integrands or with unreachable accuracy requirements for which this program is actually faster than QUADPACK.

The use of an interval-valued integrand f and limits A and B in equation (1.1) is a generalization of the definition of an interval integral given in [4]. Let $\mathbf{R}$ denote the set of real numbers, and $\mathbf{IR}$ denote the set of intervals over $\mathbf{R}$. If A and $B \in \mathbf{IR}$, and $f = [\underline{f}, \overline{f}] : \mathbf{R} \to \mathbf{IR}$ is an interval-valued function defined on the interval hull of A and B, define an integral with interval-valued limits to be

$$\int_A^B f(x)\,dx = \left\{ \left[\underline{\int}_a^b \underline{f}(x)\,dx, \overline{\int}_a^b \overline{f}(x)\,dx \right] \;\middle|\; a \in A,\ b \in B \right\},$$

where $\underline{\int}$ and $\overline{\int}$ denote the lower and upper Darboux integrals, respectively.

Usually, the limits of integration are very narrow intervals containing real numbers which are not machine numbers, such as π, or numbers which are not known precisely, but the methods of this paper apply also to integrals for which A and B are wide intervals.

There are two distinct, but related, issues to be addressed in this paper:

1. Inclusion, guarantee, or self-validation of the result, and

2. Accuracy or width of the interval.

Inclusion means that $If \in J = [c, d]$. Inclusion is achieved by capturing the truncation error terms of standard quadrature formulas as in the self-validating algorithms of Gray and Rall [7], [8], [9]. This is discussed in §§2 and 3.

High accuracy (often nearly full machine accuracy) is achieved using the accurate arithmetic of Kulisch and Miranker [10]. We use the scalar product from ACRITH [1], the notion of adaptive quadrature [2], [3], and [17], and techniques such as intersecting several estimates as discussed in §4.

2. Inclusion of Quadrature Formulas.

In this section, we assume that A and B, the endpoints of the interval of integration, are machine numbers. We denote them by a and b to distinguish them from the interval-valued limits. The case of interval-valued limits is discussed in §5.

An inclusion for the quadrature problem of equation (1.1) can be obtained from standard quadrature formulas or from Taylor polynomials. We will consider each in turn.

2.1. Inclusions using standard quadrature formulas.

The method for interval integration by use of standard formulas for numerical quadrature or Taylor series was first described by Moore [11]. To illustrate Moore's idea, consider a standard interpolatory integration *formula* of the form

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} w_i f(u_i) + c_n h \cdot \frac{f^{(p)}(\xi) h^p}{p!}, \tag{2.1}$$

where $h = b - a$, and $a < \xi < b$. Gauss and Newton-Cotes integration formulas follow this pattern [6], as does a one-term Riemann sum (with $n = 0$). Let

$$r_n f = \sum_{i=1}^{n} w_i f(u_i), \text{ and } \quad e_n f = c_n h \cdot f_p(\xi, h),$$

where $f_p(\xi, h) = \frac{f^{(p)}(\xi)h^p}{p!}$ denotes the Taylor coefficient of order p in the expansion of $f(\xi + h)$.

Let **S** denote the set of floating-point numbers available on a particular computer, and let **IS** denote the corresponding set of intervals over **S**. Let ∇ and Δ denote the monotone downward and upward roundings, respectively, from $\mathbf{R} \to \mathbf{S}$ [10]. If f is evaluated on an interval $X \in \mathbf{IS}$ using interval arithmetic and interval library functions, then the result is the *natural interval inclusion* $F(X)$ of f on X such that

$$f(X) = \{f(x) \mid x \in X\} \subseteq F(X)$$

[12], [13]. Let $X = [a, b]$, $H = [\nabla h, \Delta h]$, $W_i = [\nabla w_i, \Delta w_i]$, and $U_i = [\nabla u_i, \Delta u_i]$. Then the inclusions

$$\begin{aligned} r_n f \in R_n f &= \sum_{i=1}^{n} W_i F(U_i), \\ e_n f \in E_n f &= C_n H \cdot F_p(X, H), \text{ and} \\ If &\in R_n f + E_n f \end{aligned} \tag{2.2}$$

are guaranteed, where $F_p(X, H)$ is the natural interval extension of $f_p(\xi, h)$.

2.2. Inclusions using Taylor polynomials.

Quadrature formulas based on Taylor polynomials are well known, but they are rarely used because of the alleged difficulty of generating the series for complicated integrands. Moore [11] provides the basis for self-validating numerical integration by the use of Taylor series, although the techniques presented by him in this case are directed toward the solution of the initial-value problem for ordinary differential equations. For numerical integration, Taylor series are more appropriate than fixed quadrature formulas for interval-valued endpoints of intervals of integration, as will be discussed in §5.

Of course, one could consider problem (1.1) to be the solution $If = y(b)$ of the initial-value problem

$$y'(x) = f(x), \qquad y(a) = 0, \tag{2.3}$$

and apply Moore's methods directly. However, since $f(x)$ in (2.3) is independent of y, unlike the usual case in differential equations, it is simpler to use the capability to generate a segment of the Taylor series and the interval remainder term automatically to perform a self-validating calculation of the desired integral.

In particular, instead of expanding the solution $y(x)$ of (2.3) at $x = a$ as in the case of a differential equation, it is advantageous to expand $f(x)$ at the midpoint $c = (a+b)/2$ of the interval $X = [a, b]$ of integration. Assume that the integrand f has $p \geq 0$ derivatives on X. For $n \leq p$ and ξ between c and x,

$$\begin{aligned} f(x) &= f(c) + f'(c)(x-c) + f''(c)\frac{(x-c)^2}{2!} + \cdots + f^{(n-1)}(c)\frac{(x-c)^{n-1}}{(n-1)!} \\ &\quad + f^{(n)}(\xi)\frac{(x-c)^n}{n!} \\ &\in f(c) + f'(c)(x-c) + \cdots + f^{(n-1)}(c)\frac{(x-c)^{n-1}}{(n-1)!} + F^{(n)}(X)\frac{(x-c)^n}{n!}, \end{aligned} \tag{2.4}$$

where $F^{(n)}$ is a natural interval extension of $f^{(n)}$. Let $g(x)$ be an indefinite integral of the Taylor polynomial of degree $n-1$ given by

$$g(x) = f(c)(x-c) + f'(c)\frac{(x-c)^2}{2!} + \cdots + f^{(n-1)}(c)\frac{(x-c)^n}{n!}. \tag{2.5}$$

Then

$$\int_a^b f(x)dx \in g(x)\Big|_a^b + F^{(n)}(X)\frac{(x-c)^{n+1}}{(n+1)!}\Big|_a^b \subseteq J_n, \text{ where} \tag{2.6}$$

$$\begin{aligned} J_n &= 2\sum_{\substack{i=0 \\ i \text{ even}}}^{n-1} F^{(i)}(c)\frac{H^{i+1}}{(i+1)!} \\ &\quad + \begin{cases} \left[F^{(n)}(X)\dfrac{H^{n+1}}{(n+1)!} - F^{(n)}(X)\dfrac{H^{n+1}}{(n+1)!}\right], & \text{for } n \text{ odd}, \\ 2F^{(n)}(X)\dfrac{H^{n+1}}{(n+1)!}, & \text{for } n \text{ even}. \end{cases} \end{aligned} \tag{2.7}$$

If the series were expanded at $x = a$ instead of $x = c$, then the width of the interval remainder terms would be increased by a factor of 2^{n+1}.

Formulas (2.6) and (2.7) resemble (2.2) for ordinary quadrature rules, with the evaluation of the integrand at n points replaced by its value and the values of its first $n-1$ derivatives at a single point. If f is a polynomial of degree $n-1$ or less, then $F^{(n)}(X) \equiv [0,0]$, and only roundoff error effects the width of J_n.

3. Automatic Differentiation.

The series terms involving $F^{(n)}$ which appear in equations (2.2) and (2.6) are computed using automatic differentiation. Recurrence relations are given in [12], [13], or [16] for the Taylor coefficients for $+$, $-$, $*$, $/$, $\sqrt{}$, exp, log, sin, etc. In this application, the "point" of expansion $x = c$ and the stepsize $h = x - c$ are interval-valued, but the same recurrence relations hold.

For example, suppose that the Taylor coefficients $V_1, V_2, \ldots$ for the function $v(x)$ have already been computed. Let $u(x) = \exp(v(x))$. Then, the Taylor coefficients for u are

$$
\begin{aligned}
U_1 &= \exp(V_1) \\
U_n &= \sum_{i=1}^{n-1} \left(1 - \frac{i-1}{n-1}\right) \cdot U_i \cdot V_{n-i+1}
\end{aligned}
\tag{3.1}
$$

The expression for the integrand f is parsed into a code list [16] which is used to generate calls to appropriate subroutines for the automatic generation of Taylor coefficients. For example, consider the expression

$$f(x) = \exp(1 + x * *2).$$

The series for f expanded at C with a stepsize H is generated by a sequence of calls of the form

```
Temp1 = (C,H)    {The series for x.}
Temp2 = (1)      {Constant series.}
Call ITSQR (Temp1, Temp3)
Call ITADD (Temp2, Temp3, Temp4)
Call ITEXP (Temp3, F).
```

Subroutine ITEXP implements the recurrence relation (3.1): subroutines ITSQR and ITADD implement similar recurrence relations for $u(x) = v(x)^2$ and $u(x) = v(x)+w(x)$, respectively. The result of these computations is an array which contains the interval-valued series for f and an indication of how many terms were computed.

The Taylor coefficients of a function are maintained in a record-like structure. If "f" is the name of the function, then

LF	Index of last known nonzero term;
MF	Index of last known term;
OFL	Vector of series terms — left (lower) bound;
OFR	Vector of series terms — right (upper) bound.

This data structure allows the subroutines which implement the recurrence relations for the operators to handle constants or low degree polynomials more efficiently. The subroutines recognize at run time if the integrand is a low degree polynomial. In that case, an integration formula is used which is exact for polynomials of the appropriate degree.

Similarly, the subroutines indicate that a derivative of some order is not defined by returning a value for MF which is less than the series length which was requested. For example, the series for $f(x) = \sqrt{x}$ expanded at $C = [0,1]$ must have MF = 1 since $f'[0,1]$ does not exist.

4. Achieving High Accuracy.

In §2 and 3, we considered algorithms which give a self-validated inclusion for If. In general, the inclusions computed according to those algorithms may be very wide. In this section, we discuss ways in which high accuracy can be attained.

4.1. ACRITH scalar product.

The conventional wisdom in numerical analysis is that long scalar products like the ones in equations (2.2) and (2.7) cannot be done accurately because of the accumulation

of round-off errors. However, ACRITH [1] provides a scalar product for real- or interval-valued vectors which commits only one roundoff error. This supports a highly accurate calculation of the interval J.

The rule in equation (2.2) is a weighted average of function values, so its width is about the average of the widths of $F(U_i)$. A sufficiently fine partition of the interval of integration makes the width of the truncation error as small as we wish. Then the terms of equations (2.2) and (2.7) are managed in such a way that the rules and the truncation error terms for all subintervals in the partition are summed with a single scalar product operation. The result is that $\int f$ can in principle be evaluated with the same accuracy as f. In practice, the accuracy with which $\int f$ is evaluated is constrained by the limit M on the number of function evaluations.

The ACRITH scalar product is also used to evaluate the recurrence relations (like equation (3.1)) for series generation. For many integrands, the non-linear recurrence relations which generate their series are mildly unstable, so it is important to compute their series as accurately as possible. Any such instablility manifests itself as a relatively wide interval, but the inclusion is not compromised. Any excess width is controlled by taking a sufficiently fine partition.

4.2. Order adaptation.

The program being discussed here adapts the order of the quadrature formulas it uses and the subintervals into which it partitions the interval of integration according to the behavior of the integrand.

Equations (2.1) and (2.4) require $f^{(p)}$ to be defined on $[a,b]$. Now $p \geq 0$ is sufficient, for if $f[a,b]$ is defined, then

$$\int_a^b f(x)\,dx \in (b-a)\cdot f[a,b].$$

As discussed in §3, the subroutines for the interval Taylor operators must detect the non-existence of a derivative so that the integration routines can restrict their choice of formulas

to those for which the required derivatives exist. A more detailed discussion of the order adaptation strategies is given in [5].

4.3. Subinterval adaptation.

The usual method for improving the accuracy of a quadrature formula is to use a composite (or multi-panel) rule and a strategy for adapting the partition. This program uses a subinterval adaptation strategy similar to that used by CADRE [3] or QUADPACK [14], except that they use **estimates** for the error on each subinterval, while this program uses **guarantees**.

Consider first the composite form of a standard quadrature formula. Formula (2.1) can be interpreted as a *single-panel* rule, or as a *multi-panel* rule, meaning that a simpler formula on m points is applied k times to the corresponding number of subintervals of $X = [a,b]$, with $n \leq km$. Denote the subintervals of X by X_i, $i = 1,2,\ldots,k$, and let $h_i = w(X_i)$. Then

$$\int_{X_i} f(x)\,dx = \sum_{j=1}^{m} w_{ij} f(x_{ij}) + c_{im} h_i \cdot f_p(\xi_i, h_i),$$

holds in each subinterval. It has been shown [15] that

$$\sum_{i=1}^{k} c_{im} w(X_i) F^{(p)}(X_i) \cdot \frac{w(X_i)^p}{p!} \subseteq c_n w(X) F^{(p)}(X) \cdot \frac{w(X)^p}{p!}.$$

The width of $F_p(X, w(X))$ decreases by a factor of $w(X)^p$ as $w(X)$ becomes small. In addition, the width of $F^{(p)}(X)$ overestimates the width of $f^{(p)}(X)$ by less as $w(X) \to 0$ for $f^{(p)}(x)$ continuous [12]. Thus, the gain in accuracy obtained by calculating the error terms over smaller subintervals can be substantial.

A composite form of the quadrature formula for Taylor polynomials is handled in a similar manner. Equation (2.6) is valid even if the series for f is divergent on the interval $[a,b]$, but the width of the remainder term grows with n. In this case, low order estimates are more accurate than higher order ones, but a composite rule is often necessary to

achieve the requested accuracy. Let $a = x_0 < x_1 < \ldots < x_k = b$, and let $X_i = [x_{i-1}, x_i]$. A composite rule for Taylor polynomials can take three different forms:

i) $\sum_{i=1}^{k} \left\{ g(x)\Big|_{x_{i-1}}^{x_i} + \text{Remainder on } X_i \right\} = \sum_{i=1}^{k} J_n(\text{on } X_i)$,

ii) $\sum_{i=1}^{k} \left[g(x)\Big|_{x_{i-1}}^{x_i} \right] + \text{Remainder on } [a,b]$, or

iii) $\sum_{i=1}^{k} I_n(X_i)$.

Each of these forms has advantages. Usually, form iii) yields the narrowest result because $w(I_n(X_i)) << w(J_n(X_i))$. However, the width of the result of form iii) is the sum of $w(I_n(X_i))$, so its accuracy deteriorates as $k \to \infty$. The remainder terms in forms i) and ii) can be made arbitrarily small by choosing k sufficiently large. Hence their accuracies are limited only by the accuracy with which the series can be generated. Form ii) is most useful when only a few subintervals are necessary to achieve the requested tolerance because it avoids calculating the series for the remainder term on each subinterval. In practice, we intersect the results from form i) and form iii), since both are guaranteed to contain the correct result.

The strategy for subinterval adaptation retains all subintervals. At each step, the subinterval which makes the largest contribution to the width $w(J)$ is processed by breaking it into further subintervals. This processing continues until

(i) $w(J)$ is small enough to satisfy the accuracy requirement,

(ii) the noise inherent in function evaluation limits further reduction of $w(J)$, or

(iii) more than the maximum number M of function evaluations have been performed.

A more detailed discussion of the strategy for subinterval adaptation using guaranteed bounds for the errors is given in [5].

4.4. Intersection.

The width of the interval J can often be reduced substantially by intersecting two or more intervals which are each guaranteed to contain If. For example, one-panel Gaussian

quadrature formulas are usually given as

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} w_i f(x_i) + C_n f^{(2n)}(\xi)$$

[6], but Stroud and Secrest [18] give additional error terms

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} w_i f(x_i) + C_{n,r} f^{(r)}(\xi), \quad r = 1,2,4,\ldots. \tag{4.1}$$

Then

$$If \in \sum_{i=1}^{n} w_i f(x_i) + \bigcap_{r=1,2,4,\ldots}^{2n} C_{n,r} f^{(r)}[a,b].$$

In practice, the gain in the accuracy of J as the result of this intersection is small because the coefficients $C_{n,r}$ for the low order error terms are quite large. Occasionally, however, gains in accuracy of a full order of magnitude are achieved. The computational cost of using the error coefficients from equation (4.1) is essentially zero because the Taylor coefficients $F_k[a,b]$ for $k = 0,1,2,\ldots,2n$ are required in order to compute $F_{2n+1}[a,b]$.

A similar intersection principle can be used for Taylor polynomials. If $f^{(n)}[a,b]$ is defined for $0 \le n \le p$, and if J_n is given by equation (2.6), then $If \in J_n$, for $n = 0,1,\ldots,p$. Hence

$$If \in \bigcap_{n=0}^{p} J_n.$$

This intersection can be calculated as the corresponding interval Taylor coefficients of the integrand are generated:

$$\begin{cases} I_0 = J_0, & \text{(a Riemann sum)}, \\ I_n = I_{n-1} \bigcap J_n, & n = 1,2,\ldots,p. \end{cases}$$

This provides a means to determine the highest useful term of the Taylor expansion, since the calculation can be terminated when effective decrease in the widths of the intervals $\{I_n\}$ ceases, or when the desired tolerance is met.

5. Extension to Interval Values.

In §§2, 3, and 4, we considered the case when the endpoints of the interval of integration are machine numbers ($A = a,\ B = b \in \mathbf{S}$). Often, however, the limits are very

narrow intervals resulting from actual values which are not machine numbers or which are not known precisely. For example, consider

$$\int_{[0,0.001]}^{[5.666,5.667]} f(x)\,dx,$$

whose limits are known with an accuracy of at least 1/1000. The effect of the interval-valued endpoints is to widen the interval estimate J by an amount of the order of

$$\int_0^{0.001} f(x)\,dx + \int_{5.666}^{5.667} f(x)\,dx.$$

For some problems, this may represent a significant uncertainty.

Interval-valued limits may also be needed when all that is known about one or both limits is that they lie on some interval. For example,

$$\int_0^{[0,1]} \frac{dx}{1+x^2} = \left[\min_{[0,1]}\left\{\int_0^t \frac{dx}{1+x^2}\right\}, \max_{[0,1]}\left\{\int_0^t \frac{dx}{1+x^2}\right\}\right] = [0, \pi/4].$$

Thus the limits A and B need not be restricted to very narrow intervals.

Similar considerations apply to interval-valued integrands. The case which usually arises in practice is that the integrand f is a function not only of the independent variable x, but also depends on several parameters $c_1, c_2, \ldots, c_m$. For example, f could be a polynomial of degree $m-1$ with coefficients determined by observations,

$$f(x) = C_1 + C_2 x + \cdots + C_m x^{m-1}, \qquad C_i \in \mathbf{IS},\ i = 1, 2, \ldots, m. \tag{5.1}$$

In general, given intervals $C_1, C_2, \ldots, C_m$, it is natural to define

$$f(x; C_1, \ldots, C_m) = \{f(x; c_1, \ldots, c_m) \mid c_1 \in C_1, \ldots, c_m \in C_m\}.$$

The natural interval inclusions $F_k(X, C_1, \ldots, C_m)$ of f and its Taylor coefficients on an interval $X \in \mathbf{IS}$ are again obtainable on a computer by using interval computation and automatic differentiation. In particular, for an interval polynomial (5.1), $F_p(X, H) \equiv [0, 0]$ for $p \geq m$, just as in the real case.

The meaning of tolerance for problems with interval-valued endpoints requires some clarification. Consider the problem

$$\int_0^{[0,1]} \frac{dx}{1+x^2} = \left[\int_0^0 \frac{dx}{1+x^2}, \int_0^1 \frac{dx}{1+x^2}\right] = \left[0, \frac{\pi}{4}\right].$$

Suppose that by some algorithm we compute $J = [-0.004, 0.790]$, for example. Then $w(J) = 0.794$, although the estimate for each endpoint is in error by less than 0.005. Hence, a requested tolerance must be large enough to accommodate the uncertainties which are inherent in the problem being solved.

The cases that one or both endpoints of integration are non-degenerate intervals in ISwill now be considered. Let the endpoints be $A = [AL, AR]$ and $B = [BL, BR]$, and assume that $AR \leq BR$. Otherwise reverse the roles of A and B. Our strategy is to concentrate the uncertainty in the computations in the same places as the uncertainty in the problem, at the ends. For example,

$$\int_{[0,0.1]}^{[3.1,3.2]} f(x)\,dx = \int_{[0,0.1]}^{0.1} f(x)\,dx + \int_{0.1}^{3.1} f(x)\,dx + \int_{3.1}^{[3.1,3.2]} f(x)\,dx.$$

The resulting integrals are of four types depending on which endpoints are interval-valued:

Type RR : $A = a,\ B = b \in \mathbf{S}$: $\int_a^b f(x)\,dx$.

Type I : $A = B = [a,b] \in \mathbf{IS}$: $\int_{[a,b]}^{[a,b]} f(x)\,dx$.

Type RI : $A = a \in \mathbf{S},\ B = [a,b] \in \mathbf{IS}$: $\int_a^{[a,b]} f(x)\,dx$.

Type IR : $A = [a,b] \in \mathbf{IS},\ B = b \in \mathbf{S}$: $\int_{[a,b]}^b f(x)\,dx$.

In general, there are three cases to be considered, depending on whether A and B are

Case 1. Disjoint: $AR \leq BL$.

$$\int_A^B f(x)\,dx = \int_{[AL,AR]}^{AR} f(x)\,dx + \int_{AR}^{BL} f(x)\,dx + \int_{BL}^{[BL,BR]} f(x)\,dx,$$

and thus is the sum of integrals of types IR, RR, and RI, respectively. This case is by far the one which occurs most often.

Case 2. Overlapping: $AL \le BL \le AR \le BR$.

$$\int_A^B f(x)\,dx = \int_{[AL,BL]}^{BL} f(x)\,dx + \int_{[BL,AR]}^{[BL,AR]} f(x)\,dx + \int_{AR}^{[AR,BL]} f(x)\,dx,$$

the sum of integrals of types IR, I, and RI.

Case 3. Nested: $BL < AL \le AR \le BR$.

$$\int_A^B f(x)\,dx = -\int_{[BL,AL]}^{AL} f(x)\,dx + \int_{[AL,AR]}^{[AL,AR]} f(x)\,dx + \int_{AR}^{[AR,BR]} f(x)\,dx,$$

again the sum of integrals of types IR, I, and RI.

We are not aware of physical problems which give rise to integration problems other than Case 1, but provision for these cases adds little to the machinery which is required to handle Case 1.

The following subsections discuss the computing self-validated bounds for integrals of types RR, I, RI, and IR.

5.1. Type RR integrals: $\int_a^b f(x)\,dx$.

Here, $a, b \in \mathbf{S}$. This is the type of integral which was considered in §§2, 3, and 4. The quadrature may be performed using Gaussian, Newton-Cotes, or Taylor polynomial formulas with remainder terms to give an inclusion. An adaptive strategy and intersection of several estimates as described in §4 are used to yield high accuracy.

The basic algorithm is

1. Compute the integral on $[a, b]$;
2. Add $[a, b]$ to the list of subintervals;
3. Loop
4. Find the subinterval on which the width of the integral is largest;
5. Bisect it;
6. Compute the integral on the left subinterval;
7. Add the left subinterval to the list;

8. Compute the integral on the right subinterval;
9. Add the right subinterval to the list;
10. Compute the integral on $[a, b]$ by summing the integrals
    on all the subintervals;
11. Exit when accuracy tolerance is met;
12. Exit with warning when
13. no further improvement in accuracy is possible,
14. or M function evaluations are exceeded;
15. End loop.

Each subinterval X is maintained in a data structure of the following form:

XA	Left endpoint of the subinterval;
XB	Right endpoint of the subinterval;
OPTORD	Order of the derivative used to compute the remainder;
WIDINT	Width of the integral on this subinterval;
SINT	Interval-valued integral on this subinterval;
WEGHT	Vector of interval valued weights (Gauss and Newton-Cotes), stepsize (Taylor);
FNVAL	Vector of interval-valued function values (Gauss and Newton-Cotes), series terms (Taylor);
FNTRN	Vector of interval-valued function values (Gauss and Newton-Cotes), series terms (Taylor), including remainder terms.

At steps 1, 6, and 8, the integral is computed on the subinterval $[XA, XB]$ using one-panel versions of Gauss, Newton-Cotes, or Taylor polynomials as outlined in §§2 and

3. The weight vectors and functions are arranged in the vectors WEGHT and FNTRN, respectively, in such a way that the interval inclusion

$$J = \sum_i \text{WEGHT}_i * \text{FNTRN}_i \tag{5.2}$$

of $\int_a^b f(x)\,dx$ is computed as a single scalar product. Similarly,

$$Rf = \sum_i \text{WEGHT}_i * \text{FNVAL}_i \tag{5.3}$$

is computed as a single scalar product.

At step 13, if $J \subseteq Rf$, then the loop is exited. In this case, further reduction of the width of the truncation error cannot reduce $w(J)$. For Gauss and Newton-Cotes formulas, SINT is used only to give WIDINT. For Taylor polynomials, $\sum_i \text{SINT}_i$ is intersected with J given by equation (5.3). SINT_i is computed using the intersection principle discussed in §4.4. For a few subintervals, $\sum_i \text{SINT}_i$ is narrower than J from equation (5.2), while the situation is usually reversed for a large number of subintervals, because (5.2) uses ACRITH's accurate scalar product.

The arrangement of subintervals in the arrays listed above must be relatively straightforward to allow J to be computed by a single scalar product operation. Each iteration of the loop from step 3 to step 15 removes one subinterval from the list and replaces it with two subintervals. Subintervals are not otherwise deleted from the list. Hence, the following simple allocation scheme works: On the ith pass through the loop, the information about the left subinterval is stored in the locations previously used by its parent, and the information about the right subinterval is stored in the $(i+1)$st locations, following the already computed values. Hence, insertion requires no searching. The widest subinterval is found at step 4 by a sequential search of the array WIDINT(1..i).

By contrast, QUADPACK [14] maintains its list of pending subintervals in sorted order, so no search is necessary for the next subinterval to be processed. However, new subintervals are inserted at locations found by a sequential search, followed by changing

pointers to all following entries in the list. For each subinterval processed, the program being described here does one sequential search, while QUADPACK does two. In addition, QUADPACK uses two sets of pointer adjustments.

For integration using Taylor polynomials, the maintenance of list of subintervals is somewhat more complicated, because the program reuses the series which it has previously computed. To illustrate the ideas, consider the first execution of the loop at step 3. At that point, the list of subintervals contains only one interval, $X = [a, b]$ itself. FNTRN contains OPTORD-1 terms of the series for f expanded at $c = (a + b)/2$ and the truncation error term involving $F^{(\mathrm{OPTORD})}(X)$ given by equation (2.7). Provided that the requested tolerance exceeds the noise inherent in the function evaluation, a stepsize h can be computed which is small enough that the requested tolerance per unit step is satisfied on the interval $[c - h, c + h]$. Notice that if a relative tolerance is requested, then this requires a current estimate for J. The value of the integral on this subinterval can be computed at a cost proportional to OPTORD, instead of a cost proportional to OPTORD^2, which would be required to generate J directly by using the recurrence relations for f. Following this, the two subintervals $[a, c - h]$ and $[c + h, b]$ are processed directly. Consequently, this method breaks the subinterval of integration into three parts, rather than bisecting it.

Thus, for integration by Taylor polynomials, step 5 of the RR algorithm is replaced by

5.0′ Compute h such that the tolerance is satisfied on $[c - h, c + h]$;

5.1′ Compute the integral on $[c - h, c + h]$ from information in FNVAL;

5.2′ "left subinterval" := [XA, $c - h$];

5.3′ "right subinterval" := [$c + h$, XB].

The middle subinterval $[c - h, c + h]$ is maintained on the list of subintervals so that its contribution to J is included in the scalar product in equation (5.2). This has the helpful side effect that h can be chosen somewhat optimistically. If the choice is too optimistic, then $[c - h, c + h]$ will be selected later for further processing as the worst subinterval, at

which time it will be broken into three parts. One of the new subintervals will occupy the place of the parent interval in the list maintained, while the other two will be added to the end of the list.

A further refinement could be implemented. The stepsize h computed at step 5.0′ has the following property: On each subinterval of length $2h$ which is contained in [XA,XB], the use of a Taylor polynomial of degree OPTORD−1 yields an integral which satisfies the requested tolerance. The integration on such a subinterval can be done with half the usual work. No series for the truncation error needs to be computed because the truncation error can be bounded by using the global remainder term on [XA,XB]. If [XA,XB] can be covered by a few subintervals of length $2h$, then this could be done, and division into three parts would be needed only when the middle part is relatively small. This refinement could improve the efficiency of the program. However, the improvement would likely be modest, because the advantages of intersecting subsequent estimates on each small subinterval would be lost, and the number of subintervals added to the list would no longer be constant. The program also does not reuse function evaluations required by Gauss or Newton-Cotes formulas, although it could be modified to do so.

5.2. Type I integrals: $\int_{[a,b]}^{[a,b]} f(x)\,dx$.

Here the lower and upper limits are the same. For each Type I, RI, or IR integral, one can use standard quadrature rules with appropriate interval-valued nodes, but Taylor polynomial formulas are much more accurate.

Using the notation of §2, $g(x)$ denotes an indefinite integral of $f(x)$ (see equation (2.5)). The one-panel form of the quadrature formula for a Type I integral is then:

$$\int_{[a,b]}^{[a,b]} f(x)\,dx \subseteq g(x)\Big|_{[a,b]}^{[a,b]} + F^{(n)}([a,b])\frac{(x-c)^{n+1}}{(n+1)!}\Big|_{[a,b]}^{[a,b]}. \tag{5.4}$$

Note that the resulting interval is symmetric about 0.

The formulation of the multi-panel formula requires some care. We wish to capture

the set

$$\int_{[a,b]}^{[a,b]} f(x)\,dx = \left\{ \int_s^t f(x)\,dx \;\middle|\; s,t \in [a,b] \right\}.$$

Let $a = u_0 < u_1 < \ldots < u_k = b$, and let $U_i = [u_{i-1}, u_i]$, for $i = 1,2,\ldots,k$. Let $I(U_i) = \int_{U_i}^{U_i} f$. Now if $s,t \in U_i$, then

$$\int_s^t f \in I(U_i) \in \sum_{i=1}^{k} I(U_i),$$

while if $s \in U_i$ and $t \in U_j$ with $i < j$, then

$$\begin{aligned} \int_s^t f &= \int_s^{u_i} f + \int_{u_i}^{u_{i+1}} f + \ldots + \int_{u_{j-1}}^{t} f \\ &\in \int_{U_i}^{U_i} f + \int_{U_{i+1}}^{U_{i+1}} f + \ldots + \int_{U_j}^{U_j} f \in \sum_{i=1}^{k} I(U_i). \end{aligned}$$

Hence

$$\int_{[a,b]}^{[a,b]} f(x)\,dx \in \sum_{i=1}^{k} \int_{U_i}^{U_i} f.$$

The subintervals U_i are chosen by the same subinterval adaptation strategy used for Type RR integrals. Hence the algorithm for Type I integrals is like that for Type RR, except that the integral on each subinterval is computed using the one-panel Type I formula given by equation (5.4).

5.3. Type RI integrals: $\int_a^{[a,b]} f(x)\,dx$.

Here the left endpoint of the upper limit coincides with the lower limit. The one-panel form of the quadrature formula for a Type RI integral is then:

$$\int_a^{[a,b]} f(x)\,dx \subseteq g(x)\Big|_a^{[a,b]} + F^{(n)}([a,b])\,\frac{(x-c)^{n+1}}{(n+1)!}\Big|_a^{[a,b]}.$$

Note that the resulting interval must always contain 0 since $0 = \int_a^a f \in \int_a^{[a,b]}$.

Usually the interval $[a,b]$ is narrow enough that the one-panel formula is sufficiently accurate. Occasionally, however, a multi-panel formula is required. The multi-panel for-

mula is given in the same notation used in §5.2.

$$
\begin{aligned}
\int_a^{[a,b]} f &= \left\{ \int_a^t f \;\middle|\; t \in [a,b] \right\} \\
&= \int_{u_0}^{[u_0,u_k]} f \\
&= \int_{u_0}^{[u_0,u_1]} f \\
&\qquad \bigcup \left\{ \int_{u_0}^{u_1} f + \int_{u_1}^{[u_1,u_2]} f \right\} \\
&\qquad \bigcup \left\{ \int_{u_0}^{u_1} f + \int_{u_1}^{u_2} f + \int_{u_2}^{[u_2,u_3]} f \right\} \\
&\qquad \bigcup \quad \cdots\cdots \bigcup \left\{ \int_{u_0}^{u_1} f + \int_{u_1}^{u_2} f + \ldots + \int_{u_{k-1}}^{[u_{k-1},u_k]} f \right\}.
\end{aligned}
\tag{5.5}
$$

Thus the multi-panel formula for Type RI integrals is expressed in terms of one-panel formulas for Type RR and Type RI integrals.

Once the partition $u_1, \ldots, u_{k-1}$ is chosen, subintervals can be subdivided only with great difficulty, so this formula does not lend itself to subinterval adaptation. We observe that

$$\int_a^{[a,b]} f \subseteq \int_{[a,b]}^{[a,b]} f,$$

so the Type I algorithm (which does use subinterval adaptation) gives bounds for Type RI integrals. If those bounds are not sufficiently tight, we apply formula (5.5) using a fixed stepsize equal to the length of the smallest subinterval chosen adaptively by the Type I algorithm.

5.4. Type IR integrals: $\int_{[a,b]}^b f(x)\,dx$.

Here the right endpoint of the lower limit coincides with the upper limit. These integrals are handled in a manner similar to Type RI integrals.

6. Numerical Examples.

To give an indication of the accuracy and speed of the program reported here, we present some preliminary comparisons with the routine QAGS from QUADPACK [14].

These results are not a comprehensive test; that work is in progress. They are intended only to give an indication of the performance of the program reported here. It must be remembered that QAGS provides an **estimate** for If, while the self-validating quadrature provides an interval on which If is **guaranteed** to lie.

All of the tests reported here were conducted on an IBM 4341 using double precision arithmetic. Other tests have indicated that the self-validating program executed about 30% faster on an IBM 4361 where the ACRITH instructions are microcoded. The absolute and relative error tolerances requested were 10^{-14}.

We use some of the test problems from [14]:

Test problem 1 ([14] #2 with $\alpha = 0$): $\int_0^1 \frac{dx}{(x-\pi/4)^2+1}$.

Test problem 2 ([14] #2 with $\alpha = 5$): $\int_0^1 \frac{1024\ dx}{(x-\pi/4)^2+1/1048576}$.

Test problem 3 ([14] #3 with $\alpha = 0$): $\int_0^\pi \cos(\sin x)\, dx$.

Test problem 4 ([14] #3 with $\alpha = 1$): $\int_0^\pi \cos(2\sin x)\, dx$.

Test problem 5 ([14] #3 with $\alpha = 4$): $\int_0^\pi \cos(16\sin x)\, dx$.

Test problem 6 (similar to [14] #7 with $\alpha = 0.5$): $\int_0^1 \sqrt{|x - [0.3, 0.4]|}\, dx$.

We replaced $1/3$ by $[0.3, 0.4]$ to test a noisy function.

Test problem 7 (similar to [14] #7 with $\alpha = -0.5$): $\int_0^1 \frac{dx}{\sqrt{|x-0.375|}}$.

This is an integrable improper integral with a singularity at a machine number inside the interval of integration. This test was included to show that the self-validating program is able to detect and avoid dividing by 0.

Test problem 8 ([14] #10 with $\alpha = -0.5$): $\int_0^\pi \sqrt{\sin x}\, dx$ does not exist.

Test problem 9 ([14] #12 with $\alpha = 0$): $\int_0^1 \exp\big(20(x-1)\big) \sin x\, dx$.

Test problem 10 ([14] #12 with $\alpha = 2$): $\int_0^1 \exp\big(20(x-1)\big) \sin(4x)\, dx$.

We have only used the general purpose adaptive routine QAGS for comparison, although several of these problems are integrated faster by other routines in the QUADPACK

suite. They can also be integrated faster by a version of the self-validating program which uses special techniques for handling singularities, so we have chosen to compare only the general purpose routines here.

In order to perform a function evaluation, the expression for the integrand was accepted as a character string and parsed into a code list as discussed in §3. For each function evaluation, that code list was interpreted to generate a function value or a series as needed. In order to be comparable, the function values for QAGS were generated using the same interpreter. Other tests indicate that QAGS is about 10% faster when it is supplied with a programmed function evaluation routine (the usual method). If the program reported here is supplied with a programmed routine for series generation to replace the interpreter, it is also about 10% faster. Accordingly, for the purpose of preliminary comparisons, the use of an interpreter for function evaluation for both routines does not bias the results.

The "function evaluations" which are counted are actually *equivalent* function evaluations. The program must evaluate Taylor polynomials of different lengths. In general, the cost of generating an n-term Taylor series is $\text{Cost}(n) = \alpha n^2 + \beta n + \gamma$, where α, β, and γ are constants which depend on the integrand. Usually, α is small relative to β and γ. The number of "function evaluations" needed for the generation of an n-term series is taken as $\text{Cost}(n)/\text{Cost}(1)$. Hence the number of "function evaluations" is proportional to the CPU time needed for series generation, regardless of the number of series terms.

The self-validating program detects integrands it cannot evaluate, while QAGS does not. QAGS is capable of handling some such integrands well, but others cause it to crash. For the purposes of comparison, if the integrand could not be evaluated on the entire interval of integration, then QAGS was not called.

QAGS was not intended to handle approximate (interval-valued) integrands or endpoints. For the purposes of comparison, the problem given to QAGS is to integrate from the midpoint of A to the midpoint of B the midpoint of an interval-valued integrand. If A, B, or f are relatively wide intervals, the problem being solved by QAGS may be much

easier than the problem being solved by the self-validating program.

Test	Error Code	Absolute Error	Function Evaluations	CPU Seconds
1	0	6.4E-15	121	0.745
2	64	6.3E-08	1030	30.796
3	0	1.8E-15	683	2.517
4	0	8.4E-15	470	2.895
5	64	1.3E-10	1088	5.788
6	63	1.3E-01	677	2.975
7	61			
8	61			
9	0	6.4E-15	222	1.987
10	0	6.5E-15	239	2.675

Table 6.1. Performance of self-validating quadrature.

SVALAQ Error Codes:

0 Normal return. No errors detected.

61 Unable to evaluate the integrand.

63 Requested tolerance relaxed. Noise in function evaluation prevents reaching the requested tolerance.

64 Requested tolerance relaxed. Reached the maximum number of function evaluations.

Test	Error Code	Function Evaluations	CPU Seconds	Ratio of Times
1	0	21	0.092	8.1
2	2	1239	40.989	0.7
3	0	147	0.433	5.8
4	2	483	2.406	1.2
5	2	735	3.671	1.6
6	0	2037	4.890	0.6
7				
8				
9	0	63	0.465	4.3
10	0	63	0.566	4.7

Table 6.2. Performance of QUADPACK's QAGS.

QAGS Error Codes:

0 Normal return. No errors detected.

2 Tolerance relaxed. Roundoff error detected.

Acknowledgment.

The author wishes to thank Professor L. B. Rall for many helpful discussions.

References.

[1] ACRITH High Accuracy Subroutine Library: General Information Manual. IBM publications, GC33-6163, 1985.

[2] Carl de Boor. On writing an automatic integration algorithm, pp. 201-209 in *Mathematical Software*, ed. by John R. Rice, Academic Press, New York, 1971.

[3] Carl de Boor. CADRE: An algorithm for numerical quadrature, pp. 417-449 in *Mathematical Software*, ed. by John R. Rice, Academic Press, New York, 1971.

[4] Ole Caprani, Kaj Madsen, and L. B. Rall. Integration of interval functions. *SIAM J. Math. Anal.* **12**, no. 3 (1981), 321-341.

[5] G. F. Corliss and L. B. Rall. Adaptive, Self-Validating Numerical Quadrature, MRC Technical Summary Report No. 2815, University of Wisconsin-Madison, 1985.

[6] P. J. Davis and P. Rabinowitz. Methods of Numerical Integration, 2nd ed. Academic Press, New York, 1984.

[7] Julia H. Gray and L. B. Rall. A computational system for numerical integration with rigorous error estimation. *Proceedings of the 1974 Army Numerical Analysis Conference*, pp. 341-355. U. S. Army Research Office, Research Triangle Park, N. C., 1974.

[8] Julia H. Gray and L. B. Rall. INTE: A UNIVAC 1108/1110 program for numerical integration with rigorous error estimation. *MRC Technical Summary Report No. 1428*, University of Wisconsin-Madison, 1975.

[9] Julia H. Gray and L. B. Rall. Automatic Euler-Maclaurin integration. *Proceedings of the 1976 Army Numerical Analysis and Computers Conference*, pp. 431-444. U. S. Army Research Office, Research Triangle Park, N. C., 1976.

[10] U. W. Kulisch and W. L. Miranker. Computer Arithmetic in Theory and Practice. Academic Press, New York, 1981.

[11] R. E. Moore. The automatic analysis and control of error in digital computation based on the use of interval numbers, pp. 61-130 in *Error in Digital Computation, Vol. 1*, ed. by L. B. Rall. Wiley, New York, 1965.

[12] R. E. Moore. Interval Analysis. Prentice-Hall, Englewood Cliffs, N. J., 1966.

[13] R. E. Moore. Techniques and Applications of Interval Analysis. SIAM Studies in Applied Mathematics, 2, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

[14] R. Piessens, E. de Doncker-Kapenga, C. W. Überhuber, and D. K. Kahaner. QUADPACK: A Subroutine Package for Automatic Integration. Springer Series in Computational Mathematics, No. 1. Springer, New York, 1983.

[15] L. B. Rall. Optimization of interval computation, pp. 489-498 in *Interval Mathematics 1980*, ed. by K. L. E. Nickel, Academic Press, New York, 1980.

[16] L. B. Rall. Automatic Differentiation: Techniques and Applications. Lecture Notes in Computer Science, no. 120. Springer, New York, 1981.

[17] J. R. Rice. A metalgorithm for adaptive quadrature. *J. ACM* **22**, No. 1 (1975), 61-82.

[18] A. H. Stroud and D. Secrest. Gaussian Quadrature Formulas. Prentice-Hall, Englewood Cliffs, N. J., 1966.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 2913	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): COMPUTING NARROW INCLUSIONS FOR DEFINITE INTEGRALS	5. TYPE OF REPORT & PERIOD COVERED: Summary Report - no specific reporting period
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s): George F. Corliss	8. CONTRACT OR GRANT NUMBER(s): DAAG29-80-C-0041
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Mathematics Research Center, University of Wisconsin 610 Walnut Street Madison, Wisconsin 53705	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Work Unit Number 3 - Numerical Analysis and Scientific Computing
11. CONTROLLING OFFICE NAME AND ADDRESS: U. S. Army Research Office P.O. Box 12211 Research Triangle Park, North Carolina 27709	12. REPORT DATE: February 1986 13. NUMBER OF PAGES: 26
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Adaptive quadrature
Self-validation
ACRITH
Interval arithmetic

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We use ACRITH to implement algorithms which give narrow inclusions for definite integrals. Inclusions are obtained from familiar numerical quadrature formulas such as Gaussian or Newton-Cotes with remainder terms, or from the term-by-term integration of Taylor polynomials with remainder terms. Inclusions for the remainder terms are computed using automatic differentiation.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT - cont'd.

The inclusions are valid if the integrand or the endpoints of the interval of integration are real- or interval-valued. Interval inclusions which contain only a few machine numbers are achieved by using ACRITH's accurate scalar product, by using order and subinterval adaptation, and by using special devices such as intersection of several estimates. Numerical examples show that such narrow, *guaranteed bounds* require about four times as long to compute as the *estimates* computed by the routine QAGS from QUADPACK.

END

DTIC

6-86